

## Assignment 4-3 Polynomial Division 2

You are given two polynomials  $f(x)$  and  $g(x)$  with integer coefficients. In this problem you'll have to find out the quotient  $q(x)$  and remainder  $r(x)$  of  $f(x)$  divided by  $g(x)$ . For the sake of convenience, we define the zero polynomial as  $0x^0$ .

### Input

The input consists of  $t$  ( $30 \leq t \leq 40$ ) test cases. The first line of the input contains only positive integer  $t$ . Then  $t$  test cases follow. Each test case consists of two polynomials  $f(x)$  and  $g(x)$ . Each polynomial consists of three lines: the first line contains only one positive integer  $m$  which represents the number of nonzero terms of the polynomial; the second line consists of  $m$  nonzero integers (in the range  $[-2^{62}, 2^{62} - 1]$ ) representing the coefficients of the polynomial; the third line consists of  $m$  integers (in the range  $[0, 2^{62} - 1]$  and in decreasing order) representing the corresponding exponents of the polynomial. For example, the polynomial  $300x^{1000} + 200x^{100} + 100x^{10}$  is represented as the following three lines:

```
3
300 200 100
1000 100 10
```

You may assume that the degree of  $f(x)$  is greater than or equal to the degree of  $g(x)$ . Moreover, you may also assume that neither  $f(x)$  nor  $g(x)$  is the zero polynomial.

### Output

For each test case, you are to output a pair of three lines representing, respectively,  $q(x)$  and  $r(x)$ , in the same format as the input. You may assume that the coefficients of  $q(x)$  and  $r(x)$  can be represented by 32-bit integers, and you may also assume that  $q(x)$  and  $r(x)$  have at most 20 terms, respectively. Note that if  $r(x)$  is the zero polynomial, the output for  $r(x)$  should be as follows:

```
1
0
0
```

### Sample Input

```
2
3
1 -2 -4
3 2 0
1
```

```

4
0
3
-1 -2 4
3 2 0
4
1 2 1 -7
3 2 1 0

```

### Sample Output

```

2
1 -2
3 2
2
1 -3
1 0

```

### Requirements

In your program, a polynomial  $a_n x^{d_n} + a_{n-1} x^{d_{n-1}} + \dots + a_2 x^{d_2} + a_1 x^{d_1} + a_0 x^{d_0}$  should be represented by the following two arrays:

coefficient array: 

$n$	$n-1$	$\dots$	$2$	$1$	$0$
$a_n$	$a_{n-1}$	$\dots$	$a_2$	$a_1$	$a_0$

exponent array: 

$n$	$n-1$	$\dots$	$2$	$1$	$0$
$d_n$	$d_{n-1}$	$\dots$	$d_2$	$d_1$	$d_0$

For example, the polynomial  $-x^3 - 2x^2 + 4$  is represented by the following two arrays:

coefficient array: 

$2$	$1$	$0$
$-1$	$-2$	$4$

exponent array: 

$2$	$1$	$0$
$3$	$2$	$0$

Note that  $d_n > d_{n-1} > \dots > d_2 > d_1 > d_0 \geq 0$ , and for every  $i = 0, 1, \dots, n$ ,  $a_i \neq 0$ .

### Part of the program

Please write the functions `division` and `subtraction` to complete the following program which solves this problem. In your program, you cannot declare global

variables (including global arrays) except arraySize. Moreover, you can declare at most one array (or vector) in each of the functions division and subtraction.

```
// Polynomial division provided that the quotient and remainder have integer
coefficients
#include <iostream>
using namespace std;

// quotient = dividend / divisor; remainder = dividend % divisor provided
that
// dividendExpon[ dividendSize - 1 ] >= divisorExpon[ divisorSize - 1 ], and
// neither dividend nor divisor is the zero polynomial
void division( int dividendCoef[], long long int dividendExpon[],
               int dividendSize,
               int divisorCoef[], long long int divisorExpon[],
               int divisorSize,
               int quotientCoef[], long long int quotientExpon[],
               int &quotientSize,
               int remainderCoef[], long long int remainderExpon[],
               int &remainderSize );

// returns true if and only if polynomial1 == polynomial2
bool equal( int coefficient1[], long long int exponent1[], int size1,
            int coefficient2[], long long int exponent2[], int size2 );

// minuend -= subtrahend
void subtraction( int minuendCoef[], long long int minuendExpon[],
                  int &minuendSize,
                  int subtrahendCoef[], long long int subtrahendExpon[],
                  int subtrahendSize );

// outputs the specified polynomial
void output( int coefficient[], long long int exponent[], int size );

const int arraySize1 = 20;
const int arraySize2 = 400;

int main()
{
    int T;
    cin >> T;
    for( int t = 0; t < T; t++ )
    {
        int dividendCoef[ arraySize2 ] = {};
        long long int dividendExpon[ arraySize2 ] = {};
        int dividendSize;
        cin >> dividendSize; // input dividend
        for( int i = dividendSize - 1; i >= 0; i-- )
            cin >> dividendCoef[ i ];
        for( int i = dividendSize - 1; i >= 0; i-- )
            cin >> dividendExpon[ i ];

        int divisorCoef[ arraySize1 ] = {};
        long long int divisorExpon[ arraySize1 ] = {};
        int divisorSize;
        cin >> divisorSize; // input divisor
        for( int i = divisorSize - 1; i >= 0; i-- )
            cin >> divisorCoef[ i ];
        for( int i = divisorSize - 1; i >= 0; i-- )
            cin >> divisorExpon[ i ];

        int quotientCoef[ arraySize1 ] = {};
        long long int quotientExpon[ arraySize1 ] = {};
        int quotientSize = arraySize1;

        int remainderCoef[ arraySize2 ] = {};
        long long int remainderExpon[ arraySize2 ] = {};
        int remainderSize = dividendSize;

        // quotient = dividend / divisor; remainder = dividend % divisor
        division( dividendCoef, dividendExpon, dividendSize,
                  divisorCoef, divisorExpon, divisorSize,
```

```

        quotientCoef, quotientExpon, quotientSize,
        remainderCoef, remainderExpon, remainderSize );

    output( quotientCoef, quotientExpon, quotientSize );
    output( remainderCoef, remainderExpon, remainderSize );
}

// quotient = dividend / divisor; remainder = dividend % divisor provided
// that
// dividendExpon[ dividendSize - 1 ] >= divisorExpon[ divisorSize - 1 ], and
// neither dividend nor divisor is the zero polynomial
void division( int dividendCoef[], long long int dividendExpon[],
               int dividendSize,
               int divisorCoef[], long long int divisorExpon[],
               int divisorSize,
               int quotientCoef[], long long int quotientExpon[],
               int &quotquotientSize,
               int remainderCoef[], long long int remainderExpon[],
               int &remainderSize )
{

}

// returns true if and only if polynomial1 == polynomial2
bool equal( int coefficient1[], long long int exponent1[], int size1,
            int coefficient2[], long long int exponent2[], int size2 )
{
    if( size1 != size2 )
        return false;

    for( int i = 0; i < size1; i++ )
        if( coefficient1[ i ] != coefficient2[ i ] ||
            exponent1[ i ] != exponent2[ i ] )
            return false;

    return true;
}

// minuend -= subtrahend
void subtraction( int minuendCoef[], long long int minuendExpon[],
                  int &minuendSize,
                  int subtrahendCoef[], long long int subtrahendExpon[],
                  int subtrahendSize )
{

}

// outputs the specified polynomial
void output( int coefficient[], long long int exponent[], int size )
{
    cout << size << endl;
    cout << coefficient[ size - 1 ];
    for( int i = size - 2; i >= 0; i-- )
        cout << " " << coefficient[ i ];
    cout << endl;

    cout << exponent[ size - 1 ];
    for( int i = size - 2; i >= 0; i-- )
        cout << " " << exponent[ i ];
    cout << endl;
}

```